Implementation and optimisation of Poisson solvers on GPUs

Y. Sakai[†], S. v. Wenczowski[†]

21st January 2021

[†] Professorship of Hydromechanics, Technical University of Munich, 80333 Munich, Germany.
Email: {yoshiyuki.sakai, simon.wenczowski}@tum.de

Project objective

Adapting to the fast-moving development in modern HPC hardware is generally a tough challenge for legacy CFD codes. It is, however, also an essential step towards sustaining the code's performance and efficiency for the future. In this proposed project, therefore, we port our in-house CFD code *MGLET* (Multi Grid Large Eddy Turbulence) to GPU-accelerated heterogeneous systems.

Background

In the preceding project, we performed a SIMD optimisation to the two pressure solvers that are employed in MGLET. This was motivated by the recent trend that the modern HPC processors are equipped with ever more powerful internal vectorisation hardware to maintain the performance growth while coping with the stagnated nominal frequency issue, as well as the ever growing energy consumption concern for the HPC systems. One crucial example of such systems is SuperMUC-NG at LRZ, that is based on Intel Skylake processors being equipped with 512-bit ultrawide vector registers. By exploiting the Skylake's extensive SIMD capability, our optimised code shows up to 20% overall performance improvement even in the range of $\mathcal{O}(10^4)$ MPI processes (cf. Figure 1).

More recently, the HPC trend has been shifted to a more radical direction: The GPU-accelerated heterogeneous systems. The reason for that is, whilst the integrated vector units inside the general-purpose CPUs have significantly improved the performance, or more importantly the performance-energy ratio over the last years, there is a hard limit to the improvement and it may have been reached already. As some expert boldly claimed, the famous Moore's law may well be no more, at least for the conventional CPUs. In contrast, the GPU accelerators are equipped with a great degree of thread-level parallelism combined with the fast-access memory. Together with their excellent energy efficiency, those characteristics encourage many people to believe that the future of the exascale computing will be achieved by GPUs in one way or the other. This is the very reason why even the traditionally-CPU vendors, such as Intel, are now looking into the GPU business. In fact, such type of hardware already occupies 6 out of the top 10 on the latest TOP500 list (as of June 2020), whereas 2 other systems rely on Many Integrated Core (MIC) architectures [1]. Only the remaining 2 systems are exclusively powered by the traditional CPUs, and this downtrend for this conventional architecture type is expected to continue at least for the foreseeable future.

It is, therefore, of our essential interest, and consequently the objective of this project, to upgrade our SIMD-optimised code to be able to run on the heterogeneous systems.

At this point, it is fair to point out that the traditional CFD codes based on the incompressible Navier-Stokes equations, such as MGLET, are generally lagging behind in the GPU computing in comparison to the other HPC applications (e.g. compressible Navier-Stokes, molecular dynamics, deep learning solvers). This is due to our specific need to solve the most computationally intensive part of the code: pressure Poisson problem. The pressure problem features rigid data dependencies

of the elliptic nature, which require frequent communications between the subdomains, therefore between different GPUs where the communication bandwidth is critically limited. The above challenge elevates the code complexity necessary for a competitive computing significantly, and implies a big potential for fruitful interdisciplinary collaboration between the applicationists (ourselves) and the HPC experts, and that is exactly what we opt for. Consequently, this project will be carried out under the newly-formed collaboration between TUM, LRZ and NVIDIA. Within this collaboration, we (TUM) are responsible for actual implementation, whereas LRZ and NVIDIA will contribute by providing necessary hardware and technical advice, respectively.

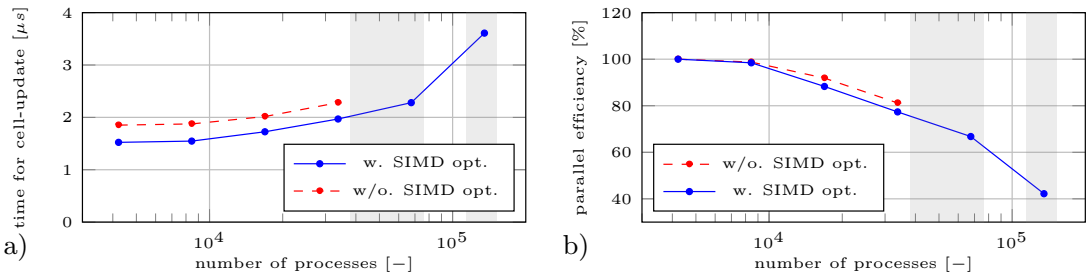

Figure 1: Weak scaling results plotted over the number of processes on SuperMUC-NG. The problem size is 6.4×10^4 cells/process over a single grid level, i.e. only intra-level boundary communication and SIP pressure solver are relevant. Changes in the background shading indicate a transition of island boundaries. For the largest case (2816 nodes), the simulation was repeated 3 times due to a considerably larger performance fluctuation than other configurations. *a)* Time for cell-update refers to the CPU-time required for one grid cell to advance one step in the time integration scheme. The value is computed from the average walltime per time step divided by the number of grid cells per process. *b)* The parallel efficiency was defined as 100% for the first plotted data point (88 nodes).

MGLET

The CFD code MGLET has been developed since the mid-1980s with the aim of simulating complex turbulent flows in academic applications/research. The software was developed at the University of the Bundeswehr Munich, Technical University of Munich, DLR Oberpfaffenhofen, the NTNU in Trondheim, Norway, and at KMUTT University and Chiang Mai University in Thailand.

MGLET is capable of performing DNS and LES of turbulent flows in arbitrary-shaped domains, which can be optionally coupled with transport of multiple scalar quantities. The code employs a finite-volume method to solve the incompressible Navier-Stokes equations for the primitive variables. Those variables are stored in a Cartesian grid with staggered arrangement and discretised in space by second-order central scheme. The time integration is done by an explicit third-order low-storage Runge-Kutta scheme [2]. The pressure computation is decoupled from the velocity computation by Chorin's projection method [3]. Accordingly, a Poisson equation is to be solved for the pressure for each Runge-Kutta sub-step. Arbitrarily curved and geometrically complex surfaces are handled by immersed boundary methods [4, 5]. A conventional domain decomposition is adopted for parallelisation, which is combined with a local grid refinement strategy. The local refinement is achieved by adding grid boxes with finer resolutions in an octree-like, hierarchical and overlapping manner, where the degree of grid refinement is determined by the grid levels [6]. This leads to two distinct types of grid-wise communications, namely: *intra-level* communication between the boundaries of neighbouring grids, and *inter-level* boundary and volume communication between adjacent grid levels.

The code is written in Fortran, and the communication between different processes is implemented via Message Passing Interface (MPI). An efficient parallel I/O strategy is implemented based on HDF5 [7]. The pressure computation in the multigrid framework is supported by the

hierarchical grid arrangement, and is accomplished by a red-black Gauss-Seidel smoother with over-relaxation (SOR) being applied at all fine grid levels. In contrast, a Strongly Implicit Procedure (SIP) solver is employed at the coarsest level [8].

GPU porting

To be ready for the future exascale HPC systems, or even for the transitional petascale systems that are on the horizon, it is necessary to upgrade MGLET by porting to GPUs. It is expected that the data flow streamlining that was performed as a part of the recent SIMD-optimisation will be a great basis for the GPU computing, thanks to its much higher degree of SIMD parallelism even in comparison to the CPUs with the ultrawide vector registers. The modern GPU's highspeed memory bandwidth will also be a significant help, since MGLET's performance is memory-bound, just like many other CFD codes. Also like many other CFD codes, the pressure solvers are the dominant hotspots in MGLET, which indicates that localised porting based on CUDA or the GPU-ready numerical libraries may be more suitable than more global, pragma-based approaches such as OpenACC or OpenMP [9].

Nevertheless, a careful decision should be made based on the performance evaluation of the respective prototypes. The standalone code snippets of the both pressure solvers from the previous project will play a key role in this step once again. Moreover, in prior to this project we extended our code snippet portfolio into a more complete framework which includes the related MPI communication routines. Extensively testing the MPI communications during the prototype phase is of a crucial importance, since we are certain that exploiting the full potential of the NVLink technology (NVIDIA's proprietary intra-GPU-node highspeed network) and the efficient MPI-communications within a GPU node is essential. For this, the technical insights available within the recently-formed three-party collaboration framework (i.e. ourselves, CFD-Lab@LRZ, NVIDIA) is there to help us. Moreover, the necessary tests will be performed on LRZ's experimental GPU server based on a NVIDIA DGX-1 node (8× Tesla V100 + Intel Xeon). After we complete actual implementation and integration of the code snippets to the production version of MGLET, we will conduct the final performance evaluation and fine tuning. In this step, we will test the completed code not only on the above experimental system, but also on a production-scale multi-node system, such as Piz Daint at CSCS or JUWELS Booster at JSC, to conclude the project.

Planned tasks

- literature review
- familialise oneself to the GPU test environment @ LRZ
- conduct a prototyping to the standalone code snippets of MGLET's pressure solvers (SIP and SOR), based on OpenACC/OpenMP (or possibly CUDA)
- perform detailed performance profilings to identify bottlenecks, analyse and implement the corresponding remedies if possible
- compare the final performance as well as the characteristics (e.g. stability, convergence) of the optimised prototypes with the ones of the PETSc solvers, which will be provided from another project that is currently in-progress
- (optional) implement the necessary GPU-GPU communication routines to enable large-scale simulations

Benefits for applicant

- access to the state-of-the-art training materials provided by NVIDIA

- in-depth supervision by experts in three different disciplines (CFD, HPC, GPU)
- close collaboration with the leading institutions both in academia and industry (LRZ, NVIDIA)

Required skills

- sufficient communication skill in English
- strong patience to work with a research code
- sufficient knowledge in UNIX/Linux system
- sufficient knowledge in HPC-relevant programming languages (e.g. Fortran, C/C++)
- experience in parallel programming (e.g. MPI, OpenMP)

Recommended skills

- experience in code optimisation
- experience in GPU programming (e.g. OpenACC, CUDA)
- basic knowledge in fluid dynamics and CFD
- basic knowledge in GIT

Contact

Dr.Ing. Yoshiyuki Sakai
Professorship of Hydromechanics
Technical University of Munich (TUM)
Arcisstr. 21, 80333 Munich, Germany
Email: yoshiyuki.sakai@tum.de

References

[1] TOP500.org. TOP500 LIST (NOVEMBER 2019), 1993-2020. https://www.top500.org/lists/2019/11/.

[2] J. H. Williamson. Low-storage Runge-Kutta schemes. *J. Comput. Phys.*, 35(1):48–56, 3 1980.

[3] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Math. Compt.*, 22(104):745–762, 1968.

[4] Nikolaus Peller, Anne Le Duc, Frédéric Tremblay, and Michael Manhart. High-order stable interpolations for immersed boundary methods. *Int. J. Numer. Methods Fluids*, 52(11):1175–1193, 2006.

[5] Nikolaus Peller. *Numerische Simulation turbulenter Str{\"o}mungen mit Immersed Boundaries Doktor-Ingenieurs*. PhD thesis, Technische Universit{\"a}t M{\"u}nchen, 2010.

[6] Michael Manhart. A zonal grid algorithm for DNS of turbulent boundary layers. *Comput. Fluids*, 33(3):435–461, 2004.

[7] The HDF Group. Hierarchical Data Format, version 5, 1997-2020. http://www.hdfgroup.org/HDF5/.

[8] Herbert L Stone. Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM J. Numer. Anal.*, 5(3):530–558, 1968.

[9] Stan Posey. Considerations for gpu acceleration of parallel cfd. *Procedia Engineering*, 61:388–391, 2013.